

# Adaptive Computation of the Klee's Measure in High Dimensions

Jérémy Barbay<sup>1</sup>, Pablo Pérez-Lantero<sup>2</sup>, and Javiel Rojas-Ledesma<sup>1\*</sup>

<sup>1</sup> Departamento de Ciencias de la Computación, Universidad de Chile, Chile  
jeremy@barbay.cl, jrojas@dcc.uchile.cl.

<sup>2</sup> Escuela de Ingeniería Civil Informática, Universidad de Valparaíso, Chile.  
pablo.perez@uv.cl.

**Abstract.** The KLEE'S MEASURE of  $n$  axis-parallel boxes in  $\mathbb{R}^d$  is the volume of their union. It can be computed in time within  $\mathcal{O}(n^{d/2})$  in the worst case. We describe three techniques to boost its computation: one based on some type of “degeneracy” of the input, and two ones on the inherent “easiness” of the structure of the input. The first technique benefits from instances where the MAXIMA of the input is of small size  $h$ , and yields a solution running in time within  $\mathcal{O}(n \log^{2d-2} h + h^{d/2}) \subseteq \mathcal{O}(n^{d/2})$ . The second technique takes advantage of instances where no  $d$ -dimensional axis-aligned hyperplane intersects more than  $k$  boxes in some dimension, and yields a solution running in time within  $\mathcal{O}(n \log n + nk^{(d-2)/2}) \subseteq \mathcal{O}(n^{d/2})$ . The third technique takes advantage of instances where the *intersection graph* of the input has small treewidth  $\omega$ . It yields an algorithm running in time within  $\mathcal{O}(n^4 \omega \log \omega + n(\omega \log \omega)^{d/2})$  in general, and in time within  $\mathcal{O}(n \log n + n\omega^{d/2})$  if an optimal tree decomposition of the intersection graph is given. We show how to combine these techniques in an algorithm which takes advantage of all three configurations.

## 1 Introduction

The KLEE'S MEASURE of a set of  $n$  axis-parallel boxes in  $\mathbb{R}^d$  is defined as the volume of the union of the boxes in the set [8]. Its computation was first posed by Victor Klee in 1977 [20], who originally considered the measure for intervals in the real line. Bentley [8] generalized the problem to  $d$  dimensions and described an algorithm running in time within  $\mathcal{O}(n^{d-1} \log n)$ . Several years later, Overmars and Yap [23] described a solution running in time within  $\mathcal{O}(n^{d/2} \log n)$ , which remained essentially unbeaten for more than 20 years until 2013, when Chan [12] presented an algorithm running in time within  $\mathcal{O}(n^{d/2})$ . We consider that, additionally, a  $d$ -dimensional domain box  $I$  is given, making the objective to compute the KLEE'S MEASURE within  $I$ .

Some special cases of this problem have been studied, such as the HYPER-VOLUME problem, where boxes are orthants of the form  $\{(x_1, \dots, x_d) \in \mathbb{R}^d \mid$

---

\* Corresponding Author

$(x_1 \leq \alpha_1) \wedge \dots \wedge (x_d \leq \alpha_d)\}$ , each  $\alpha_i$  being a real number, which can be solved in time within  $\mathcal{O}(n^{d/3} \text{polylog } n)$ ; and CUBE-KMP [2,10], when the boxes are hypercubes, which can be solved in running time within  $\mathcal{O}(n^{(d+1)/3} \text{polylog } n)$  [12]. Yildiz and Suri[26] considered  $k$ -GROUNDED-KMP, the case when the projection of the input boxes to the first  $k$  dimensions is an instance of HYPER-VOLUME. They described an algorithm to solve 2-GROUNDED in time within  $\mathcal{O}(n^{(d-1)/2} \log^2 n)$ , for any dimension  $d \geq 3$ .

The best lower bound known for the computational complexity of the KLEE’S MEASURE problem in the worst case over instances of size  $n$  is within  $\Omega(n \log n)$ , so far tight only for dimensions one and two [16] as the best known upper bound is  $\mathcal{O}(n^{d/2})$  in dimension  $d$ . Chan [11] conjectured that no ‘purely combinatorial’ algorithm computing the KLEE’S MEASURE in dimension  $d$  exists running in time within  $\mathcal{O}(n^{d/2-\varepsilon})$  for any  $\varepsilon > 0$ . He proved that if the  $d$ -dimensional KLEE’S MEASURE problem can be solved in time  $T_d(n)$ , then one can decide whether an arbitrary  $n$ -vertex graph  $G = (V, E)$  contains a clique of size  $d$  in time within  $\mathcal{O}(T_d(\mathcal{O}(n^2)))$ . The current best combinatorial algorithm for finding  $k$ -cliques in a graph, requires near- $\mathcal{O}(n^k)$  time, and hence the conjecture.

In an adaptive analysis, the cost of an algorithm is measured as a function of, not just the input size, but of other parameters that capture the inherent simplicity or difficulty of an input instance [1]. An algorithm is said to be adaptive if “easy” instances are solved faster than the “hard” ones. There are adaptive algorithms to solve classical problems such as SORTING a permutation [22], SORTING a multiset [5], computing the CONVEX HULL [19] of a set of points in the plane and in 3-space, and computing the MAXIMA of a set of  $d$ -dimensional vectors [18]. There are also adaptive algorithms for the MAXIMUM WEIGHT BOX problem [6], of particular interest since, for any dimension  $d \geq 2$ , the MAXIMUM WEIGHT BOX problem can be reduced to an instance of the KLEE’S MEASURE problem in  $2d$  dimensions.

Even though the asymptotic complexity of  $\mathcal{O}(n^{d/2})$  is the best known so far for the KLEE’S MEASURE problem [12], there are many cases which can be solved in time within  $\mathcal{O}(n \lg n)$  (see Figures 1 and 3 for some examples). Some of those “easy” instances can be mere particular cases, some others can be hints of some hidden measures of difficulty of the KLEE’S MEASURE problem.

**Hypothesis:** There are such difficulty measures that gradually separate instances of the same size  $n$  into various classes of difficulty; from easy ones solvable in time within  $\mathcal{O}(n \log n)$ , to difficult ones which the best known algorithm solves in time within  $\mathcal{O}(n^{d/2})$ .

**Results:** We describe three techniques to boost the computation of the KLEE’S MEASURE on “easy” instances, and analyze each in the adaptive model. For each technique, we identify a proper difficulty measure, which models the features which the technique is taking advantage of. The first technique is the simplest, taking advantage of degenerated instances, while the second and third ones are more sophisticated.

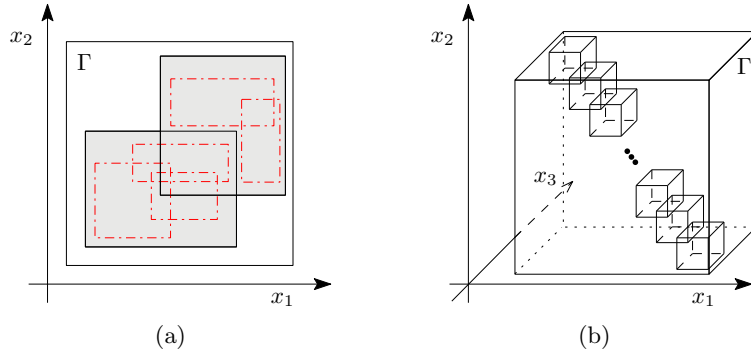


Fig. 1: Two ‘easy’ instances of the KLEE’s MEASURE problem: red dashed boxes in (a) can be removed without affecting the KLEE’s MEASURE within the domain  $\Gamma$ , yielding a much smaller instance to solve; while the instance in (b) belongs to the family of instances which intersection graph is a tree (a path in this particular case), that can be solved in time within  $\mathcal{O}(n \log n)$  by a divide-and-conquer algorithm.

The first technique (described in Section 2) is related to a classical problem in Computational Geometry: the computation of the MAXIMA of a set of vectors. A vector in a set  $\mathcal{T} \subset \mathbb{R}^d$  is called *maximal* if none of the remaining vectors in  $\mathcal{T}$  dominates it in every component. The MAXIMA of  $\mathcal{T}$  (denoted by  $M(\mathcal{T})$ ) is the set of maximal elements in  $\mathcal{T}$ . In 1985, Kirkpatrick and Seidel [18] gave an output-size sensitive algorithm for this problem, running in time within  $\mathcal{O}(n \log^{d-2} h)$ , where  $h$  is the size of the MAXIMA. We extend the concept of MAXIMA to the sets of boxes, and describe an algorithm computing the KLEE’s MEASURE in time within  $\mathcal{O}(n \log^{2d-2} h + h^{d/2}) \subseteq \mathcal{O}(n^{d/2})$ , where  $h$  denotes the size of the MAXIMA of the input set.

The second technique (described in Section 3) is based on the *profile*  $k$  of the input set, which D’Amore et al. [13] defined as the minimum, over all dimensions  $i$ , of the maximum number of boxes intersected by a same axis aligned hyperplane orthogonal to  $i$ . The algorithm described by Chan [12] to compute the KLEE’s MEASURE is in fact sensitive to a difficulty measure based on a slightly weaker definition of the profile. We improve on this by describing an algorithm to compute the KLEE’s MEASURE in time within  $\mathcal{O}(n \log n + nk^{(d-2)/2}) \subseteq \mathcal{O}(n^{d/2})$ , where  $k$  is the profile of the input set.

The third technique (described in Section 4) is based on the *treewidth* of the *intersection graph* of the input set. The intersection graph of a set of boxes is a graph  $G$  where the vertices are the boxes, and where two boxes are connected by an edge if and only if they intersect. The treewidth  $\omega$  of a graph measures how “close” the graph is to a tree. This technique yields a solution running in time within  $\mathcal{O}(n \log n + n\omega^{d/2})$  if a tree decomposition of the intersection graph of the input set is given; and a solution running in time within  $\mathcal{O}(n^4 \omega \log \omega + n(\omega \log \omega)^{d/2})$  if only the boxes are given.

In Section 5, we discuss how to compare and combine these three techniques, and in Section 6 we describe some potential directions for future work.

## 2 Maxima Filtering

Our first technique considers the MAXIMA of the input set of boxes to take advantage of instances where many boxes can be “filtered out” in small time. A box in a set  $\mathcal{B}$  is called *maximal* if none of the remaining boxes in  $\mathcal{B}$  completely contains it. The MAXIMA  $M(\mathcal{B})$  of  $\mathcal{B}$  is the set of maximal elements in  $\mathcal{B}$ . One can observe that, by definition, elements not in the MAXIMA of an input set of the KLEE’S MEASURE problem can be removed from the input set without affecting the value of the KLEE’S MEASURE.

Algorithm 1 takes advantage of this fact to compute the KLEE’S MEASURE in time sensitive to the size of the MAXIMA of the input set.

---

**Algorithm 1** maxima\_adaptive\_measure

---

**Input:** A  $d$ -dimensional domain box  $\Gamma$ , and a set of  $n$   $d$ -dimensional boxes  $\mathcal{B}$

**Output:** The KLEE’S MEASURE of  $\mathcal{B}$  within  $\Gamma$

- 1: Compute  $M(\mathcal{B})$ , the MAXIMA of  $\mathcal{B}$
  - 2: **return** SDC( $\Gamma, M(\mathcal{B})$ )
- 

Overmars [24] showed that if the MAXIMA of a set of  $n$   $d$ -dimensional vectors can be computed in time  $T_d(n)$ , then the MAXIMA of a set of  $n$  boxes can be computed in time within  $\mathcal{O}(T_{2d}(n))$ . To prove this, Overmars [24] expressed each box  $b_i = [l_{i,1}, u_{i,1}] \times \dots \times [l_{i,d}, u_{i,d}]$  as a  $2d$  dimensional vector  $\vec{b}_i = (-l_{i,1}, u_{i,1}, \dots, -l_{i,d}, u_{i,d})$ . Note that if  $b_i, b_j$  are boxes, then  $b_i$  dominates  $b_j$  if and only if  $\vec{b}_i$  dominates  $\vec{b}_j$ . We use this result to show in Lemma 1 that the KLEE’S MEASURE can be computed in running time sensitive to the size of MAXIMA of the input.

**Lemma 1.** *Let  $\mathcal{B}$  be a set of  $n$  boxes in  $\mathbb{R}^d$  and  $\Gamma$  a  $d$ -dimensional box. The KLEE’S MEASURE of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}(n(\log h)^{2d-2} + h^{d/2})$ , where  $h$  is the size of the MAXIMA  $M(\mathcal{B})$  of  $\mathcal{B}$ .*

*Proof.* Algorithm 1 achieves the bound given in the lemma: as a consequence of Overmars’ result [24], the MAXIMA  $M(\mathcal{B})$  of  $\mathcal{B}$  is computed in step one in time within  $\mathcal{O}(n \log^{2d-2} h)$  using the output-size sensitive algorithm described by Kirkpatrick and Seidel [18]; in the second step, the KLEE’S MEASURE of  $M(\mathcal{B})$ , of size  $h$ , is computed in time within  $\mathcal{O}(h^{d/2})$  using the algorithm proposed by Chan [12]. The result follows.  $\square$

Note that in the bound from the Lemma 1, the base with exponent  $d/2$  is  $h$ , instead of  $n$  as in the bound  $\mathcal{O}(n^{d/2})$  for the running time of SDC. In degenerated

instances, where  $h$  is significantly smaller than  $n$ , the bound from Lemma 1 is significantly better than  $\mathcal{O}(n^{d/2})$ .

One way to further improve this result is to remove dominated elements at each recursive call of Chan’s algorithm [12]: we discuss the difficulties in analyzing this approach in Section 6. In the next section we describe another boosting technique, which still reduces to Chan  $\mathcal{O}(n^{d/2})$ ’s algorithm, but is less focused on degenerated instances.

### 3 Profile-based Partitioning

The  $i$ -th profile  $k_i$  of a set of boxes  $\mathcal{B}$  is defined as the maximum number of boxes intersected by any hyperplane orthogonal to the  $i$ -th dimension. The profile  $k$  of a set of boxes is defined as  $k = \max_{i \in [1..d]} \{k_i\}$ . D’Amore et al. [13] showed how to compute it in linear time (after sorting the boxes in each dimension). We make the observation that Chan’s algorithm [12] for this problem is adaptive to a measure slightly different to the profile (and weaker than it); and improve on this result by describing a technique which yields a solution sensitive to the profile of  $\mathcal{B}$ .

#### 3.1 Intrinsic Adaptivity of Chan’s Algorithm

The *Simplify, Divide and Conquer* algorithm (SDC for short) proposed by Chan [12] to compute the KLEE’S MEASURE, already behaves adaptively in the sense that it runs faster on some large families of instances. Let the quasi-profile  $\kappa$  of a set of boxes be defined as  $\kappa = \max\{k_i \mid i \in [1..d]\}$ , where  $k_i$  denotes the  $i$ -th profile.

**Observation** Let  $\mathcal{B}$  be a set of boxes having quasi-profile  $\kappa$  within a domain box  $\Gamma$ . Algorithm SDC computes the KLEE’S MEASURE of  $\mathcal{B}$  within  $\Gamma$  in time within  $\mathcal{O}(n \log n + n\kappa^{(d-2)/2})$

The proof of this observation is quite technical and long. Since the result in next section subsumes this one, and the analysis there is considerably simpler, we omit the proof of this observation.

An example of the class of instances with small quasi-profile is illustrated in Figure 1b. In the next subsection, we describe a slightly modified version of the algorithm SDC which runs in time sensitive to the profile  $k$  of  $\mathcal{B}$  rather than its quasi-profile  $\kappa$ , an improvement since  $k \leq \kappa$  on all instances.

#### 3.2 Profile-based partitioning

Let  $\mathcal{B}$  be a set of boxes with profile  $k$ , and  $\Gamma$  a domain box. Given the profile  $k$  of  $\mathcal{B}$ , Algorithm 2 splits  $\Gamma$  into  $m \in \mathcal{O}(n/k)$  slabs  $\Gamma_1 \dots \Gamma_m$ , such that the measure of  $\mathcal{B}$  within  $\Gamma$  is equal to the summation of the measures of  $\mathcal{B}$  within  $\Gamma_1, \dots, \Gamma_m$ , respectively. The algorithm performs a plane sweep by one of the dimensions with the smallest profile and cuts the domain by a hyperplane every  $2k$  endpoints.

---

**Algorithm 2** split-domain

---

**Input:** A domain  $\Gamma$ , a set of  $n$  boxes  $\mathcal{B}$ , and the profile  $k$  of  $\mathcal{B}$

**Output:** A partition of  $\Gamma$  into  $m$  slabs, intersecting each one  $\mathcal{O}(k)$  boxes.

- 1: let  $i$  be a dimension where the  $i$ -th profile  $k_i$  of  $\mathcal{B}$  equals  $k$
  - 2: **for**  $j = 1, 2, \dots, m \in \mathcal{O}(n/k)$  **do**
  - 3:   let  $p \leftarrow (2k \times j)$ -th endpoint of  $\mathcal{B}$  within  $\Gamma$
  - 4:   split  $\Gamma$  into  $\{\Gamma_L, \Gamma_R\}$  by the hyperplane  $x_i = p$
  - 5:   let  $\Gamma_j \leftarrow \Gamma_L$ , and  $\Gamma \leftarrow \Gamma_R$
  - 6: **return**  $\{\Gamma_1, \dots, \Gamma_m\}$
- 

By computing the KLEE'S MEASURE of  $\mathcal{B}$  within each  $\Gamma_i$ , and summing up all those values, one can compute the KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$ .

Each of the slabs into which  $\Gamma$  is divided in Algorithm 2 can intersect at most  $\mathcal{O}(k)$  boxes of  $\mathcal{B}$ : by definition of the profile, at most  $\mathcal{O}(k)$  boxes can intersect the boundaries of the slab, and since each slab contains at most  $2k$  endpoints, no more than  $\mathcal{O}(k)$  boxes can completely lie in its interior. This can be used to bound the running time of the computation of the KLEE'S MEASURE of  $\mathcal{B}$ .

**Lemma 2.** *Let  $\mathcal{B}$  be a set of  $n$  boxes in  $\mathbb{R}^d$ ,  $\Gamma$  be a  $d$ -dimensional domain box, and  $k$  denote the profile of  $\mathcal{B}$ . The KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}\left(n \log n + nk^{\frac{d-2}{2}}\right)$ .*

*Proof.* Using Algorithm 2, one can split the domain into  $\mathcal{O}(n/k)$  slabs in linear time after sorting the input. The measure within each slab can be computed in time within  $\mathcal{O}(k^{d/2})$  using the algorithm SDC. The result follows.  $\square$

Note that again in the bound from Lemma 2, the value with  $d$  in the exponent is the profile  $k^*$ , instead of the size of the set  $n$ . Over instances with small profile  $k^*$  (like the ones in the class illustrated in Figure 1b), the bound from Lemma 2 is significantly better than the upper bound  $\mathcal{O}(n^{d/2})$  for the running time of SDC. In the next section, we describe a technique, based on the treewidth of the intersection graph of the input set, a measure that captures how “close” a graph is to a tree. This technique takes advantage of inputs where the intersection graph is of small treewidth.

## 4 Intersection Graph's Treewidth

In instances such as the one described in Figure 1b, where the intersection graph is a tree, a minor variant of SDC performs in time within  $\mathcal{O}(n \log n)$  independently of the dimension  $d$ . The concept of treewidth was discovered independently several times under different names (for a nice introduction, see Sections 10.4 and 10.5 of Kleinberg and Tardos' book [21]). Many graph problems that are NP-hard for general graphs can be solved in polynomial time for graphs with small treewidth. For example, Arnborg and Proskurowski [4] showed that for most

NP-hard problems that have linear time algorithms for trees, there are algorithms solving them in time linear in the size of the graph but exponential or super-exponential in the treewidth. They illustrated the idea with classical optimization problems involving independent sets, dominating sets, graph coloring, Hamiltonian circuits and network reliability.

In this section we describe how to generalize this behavior to instances with a more general intersection graph, taking advantage of the treewidth [21] of this intersection graph. We recall the definition and some basic results on treewidth in Section 4.1, to apply them to the computation of the KLEE'S MEASURE in Section 4.2.

#### 4.1 Preliminaries

The most widely used treewidth definition, based on *tree decompositions*, was introduced by Robertson and Seymour [25].

**Definition 1.** A *tree decomposition* of a graph  $G = (V, E)$  is a pair  $(\{X_i | i \in I\}, T = (I, F))$ , with  $\{X_i | i \in I\}$  a family of subsets of  $V$  and  $T$  a tree, such that:

- (Node coverage)  $\bigcup_{i \in I} X_i = V$ ,
- (Edge coverage) for all  $\langle u, v \rangle \in E$ , there is an  $i \in I$  with  $u, v \in X_i$ , and
- (Coherence) if  $v \in X_i \cap X_j$ , then for all  $k$  in the simple path from  $i$  to  $j$  in  $T$  we have  $v \in X_k$ .

One refers to the elements of  $I$  as *nodes*, and to the elements of  $V$  as *vertices*. The *width* of a tree decomposition  $(\{X_i | i \in I\}, T = (I, F))$  is  $\max_{i \in I} |X_i| - 1$ . A tree decomposition of  $G$  is called *optimal* if its width is the minimum width among all tree decompositions of  $G$ . The *treewidth*  $\omega$  of a graph  $G$  is the width of an optimal decomposition of itself. (see Figure 2 for an illustration of tree decompositions and treewidth).

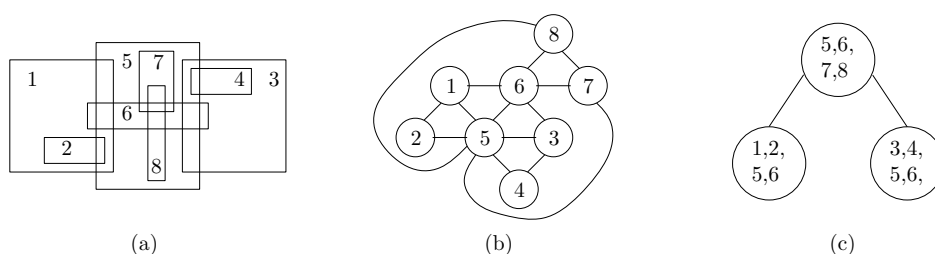


Fig. 2: Tree decomposition: (a) a set of boxes; (b) the intersection graph of the set; and (c) an optimal tree decomposition of the graph, with treewidth  $\omega = 2$

If  $T_i$  is a sub-graph of  $T$ , we use  $V_i$  to denote the vertices associated with the nodes in  $T_i$ , and  $G_{V_i}$  to denote the sub-graph of  $G$  induced by  $V_i$ .

*Property 1.* [21] Let  $k$  be a node of  $T$  and suppose that  $T - k$  has components  $T_1, T_2, \dots, T_d$ . Then, the sub-graphs  $G_{V_1 \setminus X_k}, G_{V_2 \setminus X_k}, \dots, G_{V_d \setminus X_k}$  have no vertices in common, and there are no edges between them.

A tree decomposition  $(\{X_i \mid i \in I\}, T = (I, F))$  is *nonredundant* if there is no edge  $\langle i, j \rangle$  in  $T$  such that  $X_i \subseteq X_j$ . There is a simple procedure to make any tree decomposition be nonredundant without affecting the width: if there is an edge  $\langle i, j \rangle$  in  $T$  such that  $X_i \subseteq X_j$ , one can contract the edge by ‘folding’ the node  $i$  into  $j$ ; by repeating this process as often as necessary, one ends up having a nonredundant tree decomposition.

*Property 2.* [21] Any nonredundant tree decomposition of an  $n$ -vertex graph has at most  $n$  nodes.

It is NP-hard to determine the treewidth of a given graph. Furthermore, there is no known algorithm to compute a constant-factor approximation of an optimal tree decomposition in polynomial time. The best polynomial time approximation algorithms for tree decompositions are a  $\mathcal{O}(\omega \log \sqrt{\omega})$ -factor approximation algorithm running in time within  $n^{\mathcal{O}(1)}$  described by Feige et al. [15]; and a  $\mathcal{O}(\omega \log \omega)$ -factor approximation algorithm running in time within  $(n^4 \omega \log \omega)$  described by Amir [3]. If the treewidth is known to be constant, then a  $(3\omega + 4)$ -approximation can be computed in time within  $\mathcal{O}(2^{\mathcal{O}(\omega)} + n \log n)$ , and a  $(5\omega + 4)$ -approximation can be computed in time within  $\mathcal{O}(2^{\mathcal{O}(\omega)} + n)$ , using two algorithms described by Bodlaender et al. [9] respectively.

## 4.2 An algorithm sensitive to the intersection graph’s treewidth

Here we describe an algorithm which benefits from instances that have an intersection graph with small treewidth. We will say that a set of boxes has treewidth  $\omega$  if its intersection graph has treewidth  $\omega$ .

Intuitively, consider a set of  $n$  boxes with a tree  $T$  as intersection graph. Its KLEE’S MEASURE can be computed in a divide-and-conquer fashion: reduce the problem to two sub-problems by dividing the intersection graph, via a vertex removal, into two sub-trees  $T_1, T_2$  of roughly equal sizes; solve each problem independently; and then combine their solutions. Since the intersection graph is a tree, one can always find a vertex  $v$  that divides the tree into two forests of size at most  $\lfloor n/2 \rfloor$ ; by adding  $v$  back to both forests we obtain  $T_1, T_2$  of size at most  $\lceil (n+1)/2 \rceil$ . The KLEE’S MEASURE of the original instance is the sum of the KLEE’S MEASURE of each sub-instance minus the measure of their intersection, which is the volume of the box (vertex) used to split. This procedure yields an algorithm running time within  $\mathcal{O}(n \log n)$ .

This procedure can be extended to the computation of the KLEE’S MEASURE of an instance of tree width  $\omega$ , given a tree decomposition  $T$  of its intersection graph. The following lemma shows how the solutions of two sub-problems can be combined into the general solution. If  $t$  is a node of  $T$ , we denote by  $\mathcal{B}_t$  the subset of the boxes of  $\mathcal{B}$  corresponding to the vertices within  $X_t$ .



**Lemma 3.** *Let  $T$  be a tree decomposition of the intersection graph of a set of boxes  $\mathcal{B}$ , and  $t$  be a node of  $T$  such that, when removed,  $T$  is split into two non-empty sub-forests  $F_1$  and  $F_2$ . Let  $T_L = F_1 \cup \{t\}$ ,  $T_R = F_2 \cup \{t\}$ ,  $\mathcal{B}_L = \bigcup_{l \in T_L} \mathcal{B}_l$ , and  $\mathcal{B}_R = \bigcup_{r \in T_R} \mathcal{B}_r$ . Then,  $T_L$  and  $T_R$  are tree decompositions of  $\mathcal{B}_L$  and  $\mathcal{B}_R$  respectively; and the KLEE'S MEASURE of  $\mathcal{B}$  equals the KLEE'S MEASURE of  $\mathcal{B}_L$  plus the KLEE'S MEASURE of  $\mathcal{B}_R$  minus the KLEE'S MEASURE of  $\mathcal{B}_t$ .*

*Proof.* By Property 1 we know that  $F_1$  and  $F_2$  share no vertices, and there is no edge between them. Hence, no box corresponding to a vertex in a node from  $F_1$  can intersect a box corresponding to a vertex in a node from  $F_2$ . Therefore, the intersection between  $\mathcal{B}_L$  and  $\mathcal{B}_R$  is  $\mathcal{B}_t$ . This, and the fact that the volume of a box (i.e., the KLEE'S MEASURE of a box) is a Lebesgue measure, proves that the KLEE'S MEASURE of  $\mathcal{B}_L \cup \mathcal{B}_R$  equals the KLEE'S MEASURE of  $\mathcal{B}_L$  plus the KLEE'S MEASURE of  $\mathcal{B}_R$  minus the KLEE'S MEASURE of  $\mathcal{B}_t$ . The result follows.  $\square$

Using this lemma, we can apply the procedure described above for trees to general tree decompositions, as in Algorithm 3. Lemma 4 provides an upper bound for the running time of this new solution.

---

**Algorithm 3** `tw_measure`

---

**Input:** A domain box  $\Gamma$ , and set of  $n$  boxes  $\mathcal{B}$  in  $\mathbb{R}^d$ , and an  $\rho$ -node tree decomposition  $T$  of the intersection graph of  $\mathcal{B}$

**Output:** The KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$

```

1: if  $\rho = 1$  then
2:   let  $t \leftarrow$  the only node in  $T$ 
3:   if the measure of  $t$  has not being computed before then
4:      $measures[t] \leftarrow \text{SDC}(\Gamma, \mathcal{B}_t)$ 
5:   return  $measures[t]$ 
6: else
7:   Find a node  $t \in T$  that when removed splits  $T$  into two sub-forests
      $\{F_1, F_2\}$  of sizes at most  $\lceil \rho/2 \rceil$ 
8:   let  $T_1 \leftarrow F_1 \cup \{t\}$ ,  $T_2 \leftarrow F_2 \cup \{t\}$ 
9:   Let  $\mathcal{B}_L \leftarrow \bigcup_{l \in T_L} \mathcal{B}_l$ ,  $\mathcal{B}_R \leftarrow \bigcup_{r \in T_R} \mathcal{B}_r$ 
10:  return  $\text{tw\_measure}(\Gamma, \mathcal{B}_L, T_L) + \text{tw\_measure}(\Gamma, \mathcal{B}_R, T_R) - measures[t]$ 

```

---

**Lemma 4.** *Let  $\mathcal{B}$  be a set of  $n$   $d$ -dimensional boxes. Let  $T$  be a tree decomposition of the intersection graph of  $\mathcal{B}$  with  $\rho$  nodes of sizes  $n_1, \dots, n_\rho$ , respectively. The KLEE'S MEASURE of  $\mathcal{B}$  within a given  $d$ -dimensional domain box  $\Gamma$  can be computed in time within  $\mathcal{O}(\rho \log \rho + \sum_{i=1}^{\rho} n_i^{d/2})$ .*

*Proof.* We show that Algorithm 3 runs in time within the given bound. The recursion tree corresponding to the algorithm has  $\rho$  leaves, and since at each

step the size of the problem is approximately reduced by one half, the height is within  $\mathcal{O}(\log \rho)$ . The total running time at each internal level of the tree is within  $\mathcal{O}(\rho)$ , and hence the  $\rho \log \rho$  term. Moreover, the KLEE'S MEASURE of the boxes within each leaf is computed only once, making the total time of computing the measure of the leaves to be within  $\mathcal{O}(\sum_{i=1}^{\rho} n_i^{d/2})$ . The result follows.  $\square$

When computing the KLEE'S MEASURE of a set  $\mathcal{B}$  of boxes, for whose intersection graph is given an optimal tree decomposition, the following result follows.

**Corollary 1.** *Let  $\mathcal{B}$  be a set of  $n$   $d$ -dimensional boxes, and  $T$  be an optimal non-redundant tree decomposition of the intersection graph of  $\mathcal{B}$ . The KLEE'S MEASURE of  $\mathcal{B}$  within a given  $d$ -dimensional domain box  $\Gamma$  can be computed in time within  $\mathcal{O}(n \log n + n\omega^{d/2})$ , where  $\omega$  is the treewidth of the intersection graph of  $\mathcal{B}$ .*

*Proof.* Let  $\rho$  denote the number of nodes in  $T$ . By Property 2, we know that, since  $T$  is non-redundant,  $\rho \leq n$ . Besides, each node in the tree decomposition has at most  $\omega + 1$  vertices. The result follows by replacing each  $n_i$  by  $\omega$ , for  $i = [1.. \rho]$ , in Lemma 4.  $\square$

Note that the bound  $\mathcal{O}(n \log n + n\omega^{d/2})$  in Lemma 4 is better than or equal to  $\mathcal{O}(n^{d/2})$  as long as  $\omega \leq n^{1-2/d}$ . For this bound to be achieved, an optimal tree decomposition of the intersection graph is required. This decomposition, in general, needs to be computed, and it is not known whether this can be performed in polynomial time [21].

Optimal tree decomposition of certain classes of graphs can be found efficiently. The KLEE'S MEASURE of sets with intersection graph in such classes can be computed in time depending on the treewidth. We describe two examples of such results in Corollaries 2 and 3.

**Corollary 2.** *Let  $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$  be a set of  $n$  boxes in  $\mathbb{R}^d$ ,  $\Gamma$  be a  $d$ -dimensional box, and  $G$  be the intersection graph of  $\mathcal{B}$ . The measure of the union of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}(n \log^{d-1} n + \sum_{i=1}^{\rho} n_i^{d/2})$ , where  $\rho$  is the number of connected components of  $G$ , and  $n_1, \dots, n_{\rho}$  are the sizes of the  $\rho$  connected components.*

*Proof.* An algorithm described by Edelsbrunner et al. [14] computes the connected components in time within  $\mathcal{O}(n \log^{d-1} n)$ . From the connected components one can easily obtain a tree decomposition of the graph as follows: create a node for each connected component, and add edges between the nodes until obtaining any arbitrary tree. By Lemma 4 the bound follows.  $\square$

When the profile of the input instance is  $k$ , the same bound from Lemma 2 can be achieved by using Algorithm 3, as seen in Corollary 3.

**Corollary 3.** *Let  $\mathcal{B}$  be a set of  $n$  boxes in  $\mathbb{R}^d$ ,  $\Gamma$  be a  $d$ -dimensional box, and  $k$  be the profile of  $\mathcal{B}$  within  $\Gamma$ . The KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}(n \log n + (n/k)k^{\frac{d-2}{2}})$ .*

*Proof.* We can transform  $\mathcal{B}$  into a set  $\mathcal{B}'$  with the same KLEE'S MEASURE, but with treewidth within  $\mathcal{O}(k)$ , as follows: Split the domain into  $\mathcal{O}(n/k)$  slabs using Algorithm 2. Then, for each slab, add to  $\mathcal{B}'$  the boxes in  $\mathcal{B}$  that intersect the slab, restricted to it (i.e., if a box intersects several slabs, the box will be split into multiple boxes that intersect only one slab, and such that the union of them is the original one)

A tree decomposition of the intersection graph of  $\mathcal{B}'$  with  $\mathcal{O}(n/k)$  nodes of size within  $\mathcal{O}(k)$  can be obtained as follows: create a node for each slab, and add edges between the nodes until an arbitrary tree is obtained. Since no box can intersect a box out of its slab, the tree decomposition is valid. The bound follows from applying Lemma 4.  $\square$

An approximation of the optimal tree decomposition of the input set could be used, obtaining the weaker bounds described in Corollary 4.

**Corollary 4.** *Let  $\mathcal{B}$  be a set of  $n$   $d$ -dimensional boxes, and  $\Gamma$  a  $d$ -dimensional domain box. The KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}(n^4 \omega \log \omega + N(\omega \log \omega)^{d/2})$ .*

*Proof.* The result follows by replacing the  $\omega$  term in the bound of Lemma 4 by the  $\mathcal{O}(\omega \log \omega)$ -factor approximation obtained by Amir's algorithm [3] (see the end of Section 4.1 for details).  $\square$

Note that the  $\mathcal{O}(n^4 \omega \log \omega + n(\omega \log \omega)^{d/2})$  bound for the general case is lower than  $\mathcal{O}(n^{d/2})$  as long as  $d > 8$  (because of the first term) and  $\omega \log \omega \leq n^{(d-2)/d}$ .

In the following section, we compare the techniques we have described so far and describe how to combine them.

## 5 Combining the Techniques

A low profile implies that the intersection graph has low treewidth (Corollary 3), but a low treewidth does not imply a low profile: an instance of  $n$  boxes in the class illustrated in Figure 3b has a profile within  $\mathcal{O}(n)$ , and its treewidth is one. On the other hand, the running time of the algorithm taking advantage of the profile is never worth than  $\mathcal{O}(n^{d/2})$ , which is not true for the running time of the one sensitive to the treewidth, even if an optimal tree decomposition is provided to it.

The treewidth and profile measures are independent from the size of the MAXIMA. For example, an instance of  $n$  boxes in the class illustrated in Figure 3a has a MAXIMA of size 1, and its treewidth is  $n - 1$ . With respect to the treewidth, this is a 'hard' instance, but with respect to the MAXIMA size is easy. On the contrary, an instance of  $n$  boxes in the class illustrated in Figure 3b has a MAXIMA size of  $n$ , while its treewidth is one.

Since these two measures are independent, we can combine them to obtain an algorithm sensitive to both of them, at the same time, by computing the MAXIMA of the set; and finding the KLEE'S MEASURE of the MAXIMA of the remaining graph as described in Corollary 4. This way of proceeding yields an algorithm with running time improving over the results from Lemmas 1 and 4.

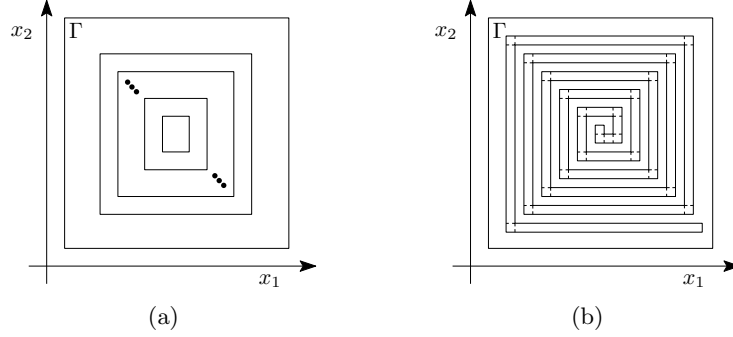


Fig. 3: Two classes of instances of the KLEE'S MEASURE problem that are “easy” or “hard” depending on the measure considered: instance (a) is easy if the size of the MAXIMA is considered, but difficult if the profile or treewidth are considered; instance (b) is easy for treewidth, but hard for both MAXIMA size and profile.

**Theorem 1.** *Let  $\mathcal{B}$  be a set of  $n$  boxes in  $\mathbb{R}^d$ ,  $\Gamma$  a  $d$ -dimensional box, and  $G$  be the intersection graph of  $\mathcal{B}$ . The KLEE'S MEASURE of  $\mathcal{B}$  within  $\Gamma$  can be computed in time within  $\mathcal{O}\left(n \log^{2d-2} h + h^4 \omega \log \omega + h(\omega \log \omega)^{d/2}\right)$ , where  $h$  is the size of the MAXIMA  $M(\mathcal{B})$  of  $\mathcal{B}$ , and  $w$  its treewidth.*

No lower bound is known for this problem with respect to these measures. In fact, we believe the results obtained here can be further improved, by considering finer versions of the measures in order to improve the analysis. We describe preliminary results in this direction in the next section.

## 6 Discussion

Each of the three boosting techniques that we analyzed can be improved, and we describe preliminary results for each technique in those directions below, as well as other lines of research.

The MAXIMA based technique (described in Section 2) yields an algorithm running in time within  $\mathcal{O}(n(\log h)^{2d-2} + h^{d/2})$ , where  $h$  is the size of the MAXIMA of the input set. This bound can be improved. For example, if instead of filtering items not in the MAXIMA only once, this is done as part of the *simplification* step of the algorithm *Simplify, Divide and Conquer* (SDC) [12], the expression for its running time becomes  $T(n) = 2T(\frac{h}{2^{2/d}}) + \mathcal{O}(n \log^{2d-2} h)$ . This running time is still within  $\mathcal{O}(n^{d/2})$  in the worst case, and also within  $\mathcal{O}(n \log^{2d-2} h + h^{d/2})$ . It is never worse (asymptotically) and it is better in many cases, but how to formally analyze this improvement is still an open question.

The profile based technique (described in Section 3) yields a solution running in time within  $\mathcal{O}\left((n - k) \log \frac{n-k}{k} + nk^{\frac{d-2}{2}}\right)$ . The algorithm SDC [12] is already adaptive to the profile of the input set. It has a limitation though: it necessarily

cycles over the dimensions in order to ensure running in time within  $\mathcal{O}(n^{d/2})$ . If there are few dimensions where the profile of the set is small, this technique performs considerably better than **SDC**. The technique could be further improved if, instead of considering an upper bound for the profile in each sub-problem, we use the exact value of the profile of the subproblem. However, it is not clear how to analyze this improvement.

The treewidth based technique (described in Section 4) yields an algorithm running in time within  $\mathcal{O}(n \log n + n\omega^{d/2})$ , if an optimal tree decomposition of the intersection graph is given; and in time within  $\mathcal{O}(n^4 \omega \log \omega + n(\omega \log \omega)^{d/2})$  if not. Note that these running time bounds are not always better than or equal to the  $\mathcal{O}(n^{d/2})$  achieved by algorithm **SDC**. The dependence on a tree decomposition is the main weakness of this approach. It is not known whether the KLEE’s MEASURE can be computed by using a treewidth-sensitive algorithm that does not depend explicitly on a tree decomposition of the intersection graph.

Finally, note that the techniques that we described focus on the structure of the instance, as opposed to the order in which the instance is given. As such, the algorithms that we described cannot beat the  $\mathcal{O}(n \log n)$  bound, even though there are instances that can still be solved in time within  $\mathcal{O}(n)$ . If one considers the order in which the input is given, for special pre-sorted inputs one can achieve the  $\mathcal{O}(n)$  bound.

## Acknowledgement

All authors were partially supported by Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003. We thank Timothy Chan for his helpful comments, and one anonymous referee from ESA 2015 for pointing out the relation between our techniques of analysis and the treewidth of the intersection graph.

## References

1. Afshani, P., Barbay, J., Chan, T.M.: Instance-optimal geometric algorithms. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 129–138 (2009)
2. Agarwal, P.K.: An improved algorithm for computing the volume of the union of cubes. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 230–239. ACM, New York, NY, USA (2010)
3. Amir, E.: Approximation algorithms for treewidth. *Algorithmica* 56(4), 448–479 (2010)
4. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k-trees. *Journal of Discrete Applied Mathematics (JDAM)* 23(1), 11 – 24 (1989)
5. Barbay, J.: From time to space: Fast algorithms that yield small and fast data structures. In: Space-Efficient Data Structures, Streams, and Algorithms (IanFest). pp. 97–111 (2013)

6. Barbay, J., Chan, T.M., Navarro, G., Pérez-Lantero, P.: Maximum-weight planar boxes in  $O(n^2)$  time (and better). *Information Processing Letters (IPL)* 114(8), 437–445 (2014)
7. Barbay, J., Pérez-Lantero, P., Rojas-Ledesma, J.: Adaptive computation of the klee’s measure in high dimensions. *CoRR* abs/1505.02855 (2015), <http://arxiv.org/abs/1505.02855>, (Last accessed on 2015-08-28.)
8. Bentley, J.L.: Algorithms for Klee’s rectangle problems. Unpublished notes (1977)
9. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A  $O(k \cdot n)$  5-approximation algorithm for treewidth. *CoRR* abs/1304.6321 (2013), <http://arxiv.org/abs/1304.6321>, (Last accessed on 2015-08-22.)
10. Bringmann, K.: An improved algorithm for Klee’s measure problem on fat boxes. *Computational Geometry: Theory and Applications (CGTA)* 45(5-6), 225–233 (2012)
11. Chan, T.M.: A (slightly) faster algorithm for Klee’s measure problem. In: *Proceedings of the Annual Symposium on Computational Geometry (SoCG)*. pp. 94–100 (2008)
12. Chan, T.M.: Klee’s measure problem made easy. In: *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. pp. 410–419 (2013)
13. d’Amore, F., Nguyen, V.H., Roos, T., Widmayer, P.: On optimal cuts of hyperrectangles. *Computing* 55(3), 191–206 (1995)
14. Edelsbrunner, H., Van Leeuwen, J., Ottmann, T., Wood, D.: Computing the connected components of simple rectilinear geometrical objects in  $d$ -space. *RAIRO - Theoretical Informatics and Applications - Informatique Thorique et Applications* 18(2), 171–183 (1984)
15. Feige, U., Hajiaghayi, M., Lee, J.R.: Improved approximation algorithms for minimum-weight vertex separators. In: *Proceedings of the annual ACM Symposium on Theory Of Computing (STOC)*. pp. 563–572. STOC ’05, ACM, New York, NY, USA (2005)
16. Fredman, M.L., Weide, B.: On the complexity of computing the measure of  $\cup_1^n [a_i, b_i]$ . *Communications of the ACM (CACM)* 21(7), 540–544 (Jul 1978)
17. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory (JCT), Series B* 16(1), 47 – 56 (1974)
18. Kirkpatrick, D.G., Seidel, R.: Output-size sensitive algorithms for finding maximal vectors. In: *Proceedings of the Annual Symposium on Computational Geometry (SoCG)*. pp. 89–96 (1985)
19. Kirkpatrick, D.G., Seidel, R.: The ultimate planar convex hull algorithm. *SIAM Journal on Computing (JC)* 15(1), 287–299 (Feb 1986)
20. Klee, V.: Can the measure of  $\cup_1^n [a_i, b_i]$  be computed in less than  $O(n \log n)$  steps? *The American Mathematical Monthly (AMM)* 84(4), 284–285 (1977)
21. Kleinberg, J., Tardos, E.: *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005)
22. Moffat, A., Petersson, O.: An overview of adaptive sorting. *Australian Computer Journal (AJC)* 24(2), 70–77 (1992)
23. Overmars, M.H., Yap, C.: New upper bounds in Klee’s measure problem. *SIAM Journal on Computing (JC)* 20(6), 1034–1045 (1991)
24. Overmars, M.H.: On the equivalence of rectangle containment, rectangle enclosure and ECDF-searching. Technical report, University of Utrecht, Department of Computer Science (01 1981)

25. Robertson, N., Seymour, P.: Graph minors. ii. algorithmic aspects of tree-width 7(3), 309 – 322 (1986)
26. Yildiz, H., Suri, S.: On Klee’s measure problem for grounded boxes. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 111–120. ACM, New York, NY, USA (2012)